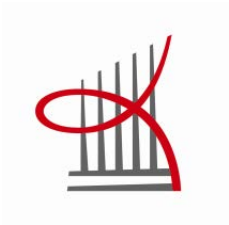


SUORITUSKYKYTESTITYÖKALUN GRAAFINEN KÄYTTÖLIITTYMÄ

Jonas Nyman

Opinnäytetyö
Toukokuu 2012
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka

TAMPEREEN AMMATTIKORKEAKOULU
Tampere University of Applied Sciences

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Ohjelmistotekniikan suuntautumisvaihtoehto

NYMAN, JONAS:
Suorituskykytestityökalun graafinen käyttöliittymä

Opinnäytetyö 51 sivua, josta liitteitä 14 sivua
Toukokuu 2012

Nykyään yhä useammat työt tehdään tietokoneohjelmien avulla. Monista ohjelmista puuttuu kuitenkin edelleen graafinen käyttöliittymä, jonka takia ohjelman käyttäminen voi olla vaikeaa. Tekstikomennoilla toimivaa ohjelmaa käyttäessä käyttäjän on tunnettava ohjelman syntaksi.

Graafisen käyttöliittymän avulla käyttäjä näkee visuaalisesti esitettynä kaikki mahdolliset toiminnot ja komentojen antaminen toimii pääsääntöisesti hiirtä käyttäen. Graafinen käyttöliittymä mahdollistaa paremman vuorovaikutuksen käyttäjän kanssa, jolloin käyttäjä saa paremman kuvan ohjelman toiminnasta.

Tämän opinnäytetyön tavoitteena oli luoda jo valmiille konsoliohjelmalle graafinen käyttöliittymä. Konsoliohjelman omistaa Nokia Siemens Networks. Monilla uusilla käyttäjillä oli vaikeuksia omaksua konsoliohjelman toiminnallisuuksia ja useat käyttäjät ovat toivoneet graafista käyttöliittymää. Tein yhteistyötä kehittäjien ja käyttäjien kanssa, kun suunnittelin ja loin ohjelmalle graafisen käyttöliittymän. Lopputulos on auttanut nykyisiä ja uusia käyttäjiä tekemään työnsä tehokkaammin.

Avainsanat graafinen käyttöliittymä, ohjelmistokehitys, testityökalu

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applies Sciences
Degree Programme in ICT Engineering
Option of Software Engineering

NYMAN, JONAS:
Graphical user interface for a performance test tool

Bachelor's thesis 51 pages, appendices 14 pages
May 2012

Most of the work today is done with the help of computer programs. Many programs, however, still lacks a graphical user interface, which is why the program may be difficult to use. When using a program with text commands, the user must be familiar with the program's syntax.

A graphical user interface allows the user to see visually all the possible functions and the commands are mostly given with the mouse. A graphical user interface allows a better interaction with the user and gives the user a better picture of the programs functionality.

The purpose of this thesis was to create a graphical user interface to a console program. The console program is owned by Nokia Siemens Networks. Many new users thought it was difficult to understand the console programs functionality and many had hoped for a graphical user interface. I worked closely with the developers and users, when I designed and created a graphical user interface for the program. The graphical user interface has helped the existing and new users to do their jobs more efficiently.

Keywords graphical user interface, software development, testing tool

ESIPUHE

Työn tilaajana oli Nokia Siemens Networks. Olen kyseiselle yritykselle tehnyt töitä aikaisemmin ja he pyysivät minua tekemään graafisen käyttöliittymän heidän käyttämään ohjelmaan. Ohjelman on tarkoitus helpottaa suorituskykytestaajien työtä sekä Suomessa että Puolassa. Tehtävä oli minulle täysin uudenlainen, koska en ole aiemmin tehnyt verkkosivuja, puhumattakaan Linux-terminaalin ja Php:n yhdistämistä. Työ opetti minulle paljon uusia tekniikoita.

Toukokuu 2012

Jonas Nyman

SISÄLLYS

Termien selitykset

Agile	Ketterä ohjelmistokehitys
Apache	Verkkopalvelin, lähettää HTML-sivut käyttäjälle
Bash	Komentotulkki unix-tyyppisissä käyttöjärjestelmissä
Fedora	Red Hatin Linux variantti
GUI	Graphical user interface, suomeksi graafinen käyttöliittymä
HTML	Tulkattava kieli verkkosivuissa
JavaScript	Ohjelmointikieli nettisivuissa, toimii käyttäjän koneessa
Kernel	Käyttöjärjestelmän ydin, yhdistää ohjelmiston ja raudan
Linux	Käyttöjärjestelmä, Unix-klooni
Lähdekoodi	Ohjelman luettava koodi
MasterCLI	Sama kuin NAPET, ajettavan ohjelman nimi on MasterCLI
NAPET	Suorituskykytestityökalu millä graafinen käyttöliittymä tehtiin
NetAct	NSN:n luoma mobiiliverkon hallintaohjelma
Paketti	Paketti, joka sisältää ohjelman
PHP	Skriptikieli verkkosivuissa, pyörii palvelimella
Red Hat	Linux variantti
Repository	Palvelimelle tallennetut tiedostot ja historia niiden versioista
RHEL	Red Hat Enterprise Linux, Red Hatin palvelinversio
Rpm-paketti	Red Hat Package Manager: Red Hatin paketinhallintaohjelma
Screen	Sallii ohjelman pyörittämisen taustalla. Ohjelma ei sulkeudu ennen kuin screen lopetetaan.
Selain	Ohjelma, joka tulkitsee HTML:ää ja esittää sen graafisesti
Skripti	Joukko käskyjä, joita kone suorittaa järjestyksessä
Syntaksi	Ohjelmistotekniikassa varattujen sanojen ja lauseiden tunnistus
SSH	Salattu konsoliyhteys Unix-ympäristöön
Unix	Käyttöjärjestelmä
Virtuaalikone	Ohjelmallisesti toteutettu tietokone, jossa voidaan ajaa ohjelmia kuin aidossa koneessa
Yum	Yellow Dog Updates, Modified: paketinhallintaohjelma rpm-pohjaisille Linuxeille

1 JOHDANTO

Lähes kaikki ohjelmat ovat nykypäivänä graafisia, sillä ne helpottavat hahmottamaan ohjelman toiminnallisuutta. Komentokehotteessa pyörivät ohjelmat ovat yhtä toimivia kuin graafisetkin, mutta vaikeampia käyttää, koska käyttäjän on tunnettava ohjelman syntaksi.

Toisen luvun aiheena on Napet, joka on Nokia Siemens Networks:in luoma ohjelma, jolle minä loin graafisen käyttöliittymän. Luku kertoo myös lyhyesti miksi ohjelma on niin tärkeä Nokia Siemens Networksille.

Opinnäytetyöni seuraa ohjelmiston kehitystä alusta loppuun. Ohjelmistokehitys aloitettiin suunnittelulla, ohjelmointikielen päättämisellä ja vaatimuksien määrittelyllä. Ohjelman kehityksen aikana tein paljon yhteistyötä käyttäjien ja kehittäjien kanssa, jotta ohjelma sisältäisi kaikki tarvittavat ominaisuudet. Kolmannessa luvussa kerrotaan tarkemmin työskentelytavasta ketterillä menetelmillä.

Neljännessä luvussa hahmotetaan eri ohjelmointikielten vahvuudet ja puutteet. Näiden tietojen ja työn tilaajan vaatimuksien perusteella päätettiin käytettävä ohjelmointikieli, jolla graafinen käyttöliittymä luotaisiin.

Viides luku käsittelee työn eri vaiheita alusta loppuun. Luku näyttää miten ohjelma kehittyi kohti graafista lopputuotetta. Samalla kerrotaan myös mitä viikoittaisissa palavereissa päätettiin; esimerkiksi alussa ohjelman ulkonäöstä ei ollut tietoa, joten se kehittyi viikon välein.

Jotta ohjelmaa pystyisi käyttämään, piti tehdä tutkimusta kuinka vaadittavat paketit saadaan asennettua Linuxiin. Tätä asiaa käsitellään kuudennessa luvussa. Ongelmaksi tuli vanha Linux-versio, jota ei enää tueta ja ohjelmien saaminen toimimaan vanhaan versioon oli varsin haasteellista. Luvussa tutkitaan myös mahdollisuutta käyttää uudempaa Linux-versiota, jolloin ohjelmistotuki olisi huomattavasti parempi. Liite 2 sisältää ohjelman täydellisen asennusohjeen.

Seitsemäs luku käsittelee erilaisia testausmenetelmiä ja miten testaus otettiin mukaan projektiin. Testaus on yksi tärkeimmistä osista ohjelmistokehityksessä, joten se kuului olennaisena osana projektia.

Kahdeksannessa luvussa käsitellään versionhallintaa. Versiohallinnan avulla ohjelmistokehittäjät pystyvät helpommin palaamaan vanhempaan koodiin, jos uudet muutokset eivät toimi halutulla tavalla. Työssä tutkitaan erilaisia vaihtoehtoja versiohallinnalle ja miksi tähän työhön valittiin tietty vaihtoehto.

Viimeinen luku käsittelee ohjelman julkaisua käyttäjille. Lisäksi pohditaan kuinka julkaisu olisi voitu hoitaa paremmin.

2 TYÖN TAVOITE

Nokia Siemens Networks työllistää useita ohjelmistotestaajia yhtiön omia ohjelmistoprojekteja varten. Yksi testaajien käyttämä ohjelma on Napet, jonka avulla testaajat pystyvät testaamaan NetAct-nimisen ohjelman suorituskykyä suurilla työkuormilla. Napet toimii ainoastaan Linux-terminaalissa, ilman minkäänlaista graafista käyttöliittymää. Ohjelman toiminnallisuuden opettaminen uusille käyttäjille on ollut vaikeaa ja tämän takia minua pyydettiin luomaan Napet GUI. Napet GUI:n avulla on tarkoitus saada Napet:in toiminnallisuus yksinkertaisemmaksi ja käyttäjäystävällisemmäksi.

2.1 NetAct

Mobiiliverkot ovat tänä päivänä erittäin suuria ja niiden hallintaa varten vaaditaan tehokkaita ohjelmia. NetAct on Nokia Siemens Networksin luoma Linux-ympäristössä toimiva verkonhallintaohjelma. Monet suuret mobiiliyhtiöt ympäri maailmaa käyttävät NetAct:ia hallitakseen mobiiliverkkojaan. (TmForum: Conformance Certification Nokia Siemens Networks NetAct 2012.)

3 TYÖSKENTELYTAPA

Työ tehtiin agile-tyylisesti. Kerran viikossa pidettiin palaveri, missä suunniteltiin tavoitteet seuraavaa viikkoa varten. Samalla kun toiminnallisuutta suunniteltiin, pohdimme myös ulkonäköä sekä sitä, kuinka uudet ominaisuudet vaikuttavat nykyiseen ulkonäköön ja miten sitä tulisi muokata.

3.1 Ketterä ohjelmistokehitys

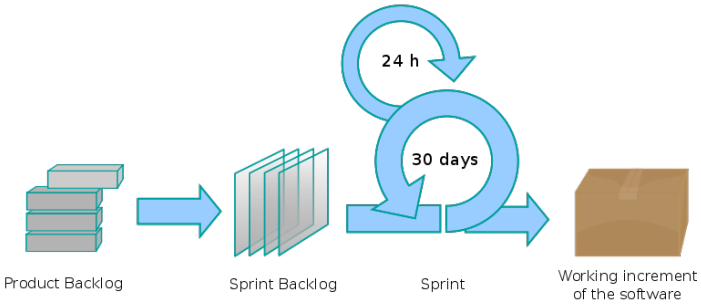
Agile-menetelmä, suomeksi ketterä ohjelmistokehitys, on joukko ohjelmistotuotantoprojekteissa käytettäviä menetelmiä, joiden avulla minimoidaan riskejä jakamalla ohjelmistokehitys lyhyisiin pätkiin. Tässä työssä työ oli pilkottu viikon mittaisiksi osiksi, jolloin kyettiin paremmin seuraamaan ohjelman kehitystä ja nopeasti puuttua epäkohtiin. (Agile Finland: Ketterät Menetelmät 2012.)

3.2 Scrum

Scrum on ehkä suosituin ketterä ohjelmistokehitysmenetelmä. Scrummin tavoitteena on kehittää tuote jokaisen kehitysjakson jälkeen täydellisemmäksi ja lähemmäksi lopputulosta. Kehitysjaksoa kutsutaan sprintiksi, joka on 1-4 viikon mittainen jakso, jonka aikana tuotetaan mahdollinen julkaisukelpoinen tuote. Ennen jokaisen sprintin alkua pidetään suunnittelupalaveri, jossa jaetaan tehtävät ja tavoitteet sprinttiä varten. Sprintin jälkeen pidetään sprinttikatselmus, missä esitellään saavutukset tuoteomistajalle. Ennen seuraavan sprintin aloitusta pidetään palaveri missä keskustellaan mikä sujui edellisessä sprintissä hyvin ja mitä voitaisiin parantaa seuraavassa sprintissä.

Kuviossa 1 näemme scrum-prosessin. Prosessi alkaa product backlogista, suom. työlista, missä tuotteen omistaja määrittää vaatimuksensa lopullista tuotetta varten. Työlistaan määritetään aika, jota tarvitaan työtä varten. Sprint kestää 1-4 viikkoa; kuvassa sprint kestää 30 päivää ja kerran päivässä pitävät scrum-tiimit keskenään palaverin, jota johtaa scrum-masteri. Masteri huolehtii, että työt suoritetaan ja jakaa lisätehtäviä tarpeen vaatiessa. Jokaisen sprintin jälkeen pyritään saamaan uusi versio

ohjelmasta. Kun ohjelma vastaa työlistan vaatimuksia, julkaistaan lopullinen tuote. (Scrum: Improving The Profession of Software Development, 2012.)



Kuvio 1: Scrum prosessi (Scrum: Improving The Profession of Software Development 2012)

3.3 Työn toteutus

Ohjelma luotiin ja testattiin virtuaalikoneessa VMware player – nimisellä ohjelmalla. Virtuaalikoneessa oli asennettuna Fedora 15 Linux–käyttöjärjestelmä, koska vaatimuksena oli, että ohjelma toimii Red Hat–pohjaisessa käyttöjärjestelmässä. Ohjelma toteutettiin PHP:lla, HTML:llä, JavaScriptillä sekä bash skriptauskielellä.

Työtä tehtiin 3 päivää viikossa 5 kuukauden ajan. Tähän aikaan kuului ohjelman suunnittelu, ohjelmointi ja dokumentointi.

4 OHJELMOINTIKIELEN VALITSEMINEN

Työn alussa tuli määritellä kuinka ohjelma toteutetaan. Vaatimuksena oli, että graafista käyttöliittymää voi käyttää missä tahansa, kaikissa käyttöjärjestelmissä, ilman ylimääräistä vaivaa käyttäjältä. Ylimääräiseksi vaivaksi laskettiin esim. kolmannen osapuolen ohjelmien asennus. Näiden vaatimuksien jälkeen vaihtoehdoiksi jäivät seuraavat ohjelmointikielet: C++ Qt-kirjastoilla, Java sekä PHP ja HTML.

Työ oli aluksi tarkoitus toteuttaa Qt:lla, mutta työn tilaaja oli huolissaan Qt:n elinkaaresta. Qt:ta kehittää Nokia, joka ilmoitti vuonna 2010, että lopettaa Symbianin, Maemon ja Meegon kehitystä, joissa käytettiin Qt:ta. Tämä oli suuri takaisku Qt:lle, jolloin sen tulevaisuus muuttui epävarmaksi. Qt-sovelluksen toimivuutta ei myöskään voida taata jokaiseen Linux-versioon. Jälkimmäinen huoli koski myös Javaa, jolloin ainoaksi vaihtoehdoksi jäi PHP ja HTML. PHP vaatii ainoastaan tukea palvelimelta ja HTML toimii jokaisessa ympäristössä, johon on asennettu selain. Ainoa poikkeus oli Internet Explorer, joka ei täysin tue yhteisiä standardeja, joita kaikki muut selaimet tukevat. Alussa päätimme, että alustavasti käyttöliittymä ei tue Internet Exploreria, koska tämä vaatisi enemmän aikaa työn tekemiseen.

4.1 C++ Qt-kirjastoilla

Qt on alustariippumaton kehitysympäristö graafisille ohjelmille. Kirjastoa kehitti alun perin ohjelmistoyritys Trolltech, jonka Nokia osti vuonna 2008. Qt:lla on sisäänrakennettu tuki C++:lle, mutta Qt tukee myös muita kieliä kuten C#, Java, Python, Ruby ja PHP. (Nokia Qt: Cross-platform application and UI framework 2012.)

4.2 Java

Java on laitteistoriippumaton oliopohjainen ohjelmointikieli, jonka on kehittänyt Sun Microsystems. Ohjelman syntaksi muistuttaa kovasti C:tä ja C++:aa, mutta yksinkertaisemmalla oliotyylillä.

Java on suunniteltu toimimaan kaikissa ympäristöissä ilman, että ohjelma tarvitaan kääntää uudestaan jokaista ympäristöä varten erikseen. Tämä on mahdollista sillä, että

lähdekoodi ei käännetä suoraan konekieleksi, kuten normaalit ohjelmat, vaan tavukoodiksi, joka suoritetaan virtuaalikoneessa missä Java-kääntäjä tulkitsee koodia. Tämä johtaa siihen että Java-ohjelmat ovat paljon raskaampia kuin normaalit ohjelmat. (Java: Java Online 2012.)

4.3 PHP ja HTML

HTML on kuvauskieli, ei ohjelmointikieli, jolla luodaan verkkosivuja. HTML on lyhennetty englanninkielen sanoista Hyper Text Markup Language, suom. hypertekstin merkintäkieli. Internet-selaimet tulkitsevat verkkosivujen HTML-koodia ja luovat sen perusteella sivun sisällön. (W3Schools: HTML Intro 2012.)

PHP on skriptauskieli, jota käytetään verkkosivujen luonnissa. PHP ei itse kykene luomaan verkkosivua, jolloin se käyttää hyödykseen HTML:ää. PHP on verkkopalvelimella toimiva kieli. Tämä tarkoittaa että funktioiden laskutoimenpiteet suoritetaan palvelimella, ei käyttäjän koneella (PHP: PHP Hypertext Proprocessor 2012). Käyttäjän koneella toimiva JavaScript käydään läpi luvussa 3.4.

4.4 JavaScript

JavaScript on monesti verkkosivuissa käytetty skriptikieli. PHP:tä ja JavaScriptiä käytetään yleensä yhdessä molempien rajoituksien takia. PHP on rajoitettu toimimaan ainoastaan palvelimen puolella, kun taas JavaScript suoritetaan käyttäjän koneella. Eri laskutoimenpiteet, tietokantakyselyt ym. joudutaan suorittamaan verkkopalvelimella. Tätä varten käytetään PHP:tä. PHP ei kykene esimerkiksi selvittämään käyttäjän käyttämää selainta, tätä varten tarvitaan JavaScriptiä. (The JavaScript source: JavaScript 2012.)

Ehkä suurin ero PHP:ssä ja JavaScriptissä on se, että sivun latauduttuaan ei PHP:llä kyetä enää tekemään sivuun muutoksia. JavaScript kykenee pyörimään selaimessa jatkuvasti, jos niin on määritetty.

JavaScriptiä käytettiin myös tässä työssä, koska muutamia ominaisuuksia ei pystytty toteuttamaan PHP:ssä.

5 OHJELMAN KEHITYS

Palaveri liittyen ohjelman suunnitteluun pidettiin kerran viikossa. Palavereissa käsiteltiin kuinka edellisen viikon lisäykset ovat toteutuneet. Jos jokin ei vaikuttanut hyvältä ratkaisulta, se muutettiin tai poistettiin kokonaan. Seuraavaksi siirryttiin eteenpäin ja päätettiin mitä ominaisuuksia tulisi seuraavaa viikkoa varten toteuttaa sekä kuinka ne tulisi näyttää graafisesti.

5.1 Eri vaiheet

Ohjelmaa kehitettiin viikon välein; alla ovat tärkeimmät vaiheet näkyvissä. Viikot ovat merkittynä työviikkoina, ei kalenteriviikkoina.

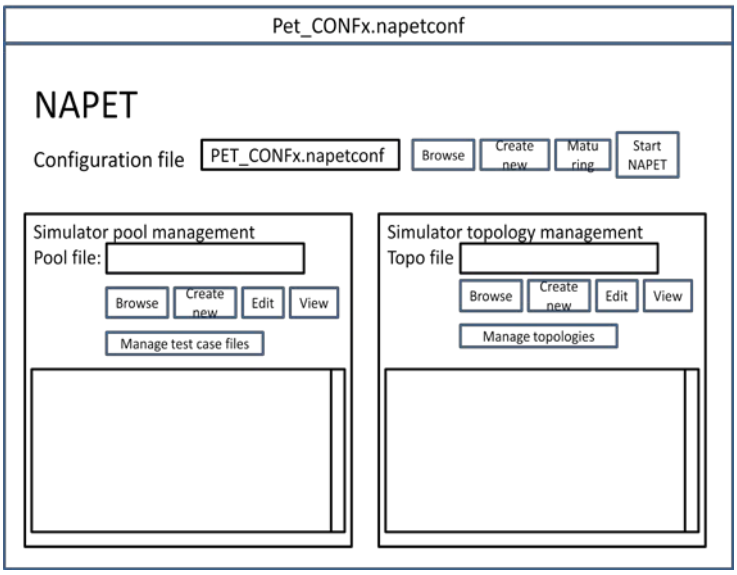
5.1.1 Viikko 1

Ensimmäisen viikon palaverissa suunnittelimme tuotteen omistajan ja muiden asiantuntijoiden kanssa ohjelman sisällöstä ja toiminnallisuudesta. Ohjelman on tarkoitus ohjata Napet-nimistä ohjelmaa, jota kutsutaan myös nimeä masterCLI, mistä ohjelma sai nimensä Napet GUI. Napet GUI ottaa ssh-yhteyden koneeseen johon Napet on asennettuna. Kun käyttäjä painaa jotain komentoa Napet GUI:ssa, lähetetään vastaava komento tekstikomentona Napetille ssh:n ylitse.

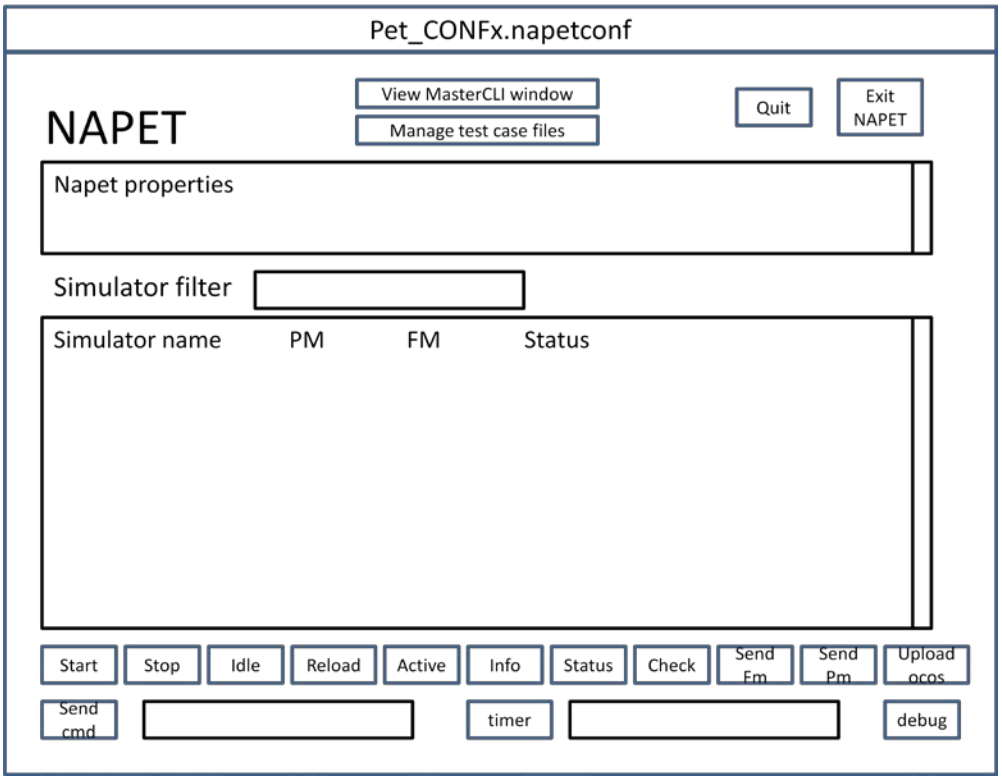
Kuvio 2 ja kuvio 3 ovat tuotteen omistajan tekemät kuvat mahdollisesta lopputuloksesta. Kuvio 3 on yhdessä tehty käyttötapauskaavio tulevasta tuotteesta.

Kuvio 2 esittää ohjelman etusivua. Tässä käyttäjä antaa ohjelmalle xml-tiedostoja, joiden avulla Napet osaa luoda simulaattoreita. Simulaattorit simuloivat NetActia, joka on Nokia Siemens Networksin luoma ohjelma. Simulaattorit palauttavat eri arvoja, joita testaajat käyttävät hyödykseen. Käyttäjän painettua etusivulla Start NAPET–nappia, käynnistetään Napet ssh:n yli, parametreina käyttäjän antamat xml-tiedostot. Napetin käynnistettyä kysytään Napetilta tietoa simulaattoreista: nimi, PM (mittausarvo), FM(hälytysarvo) ja status, joka kertoo missä tilassa simulaattori on (ON/OFF/IDLE).

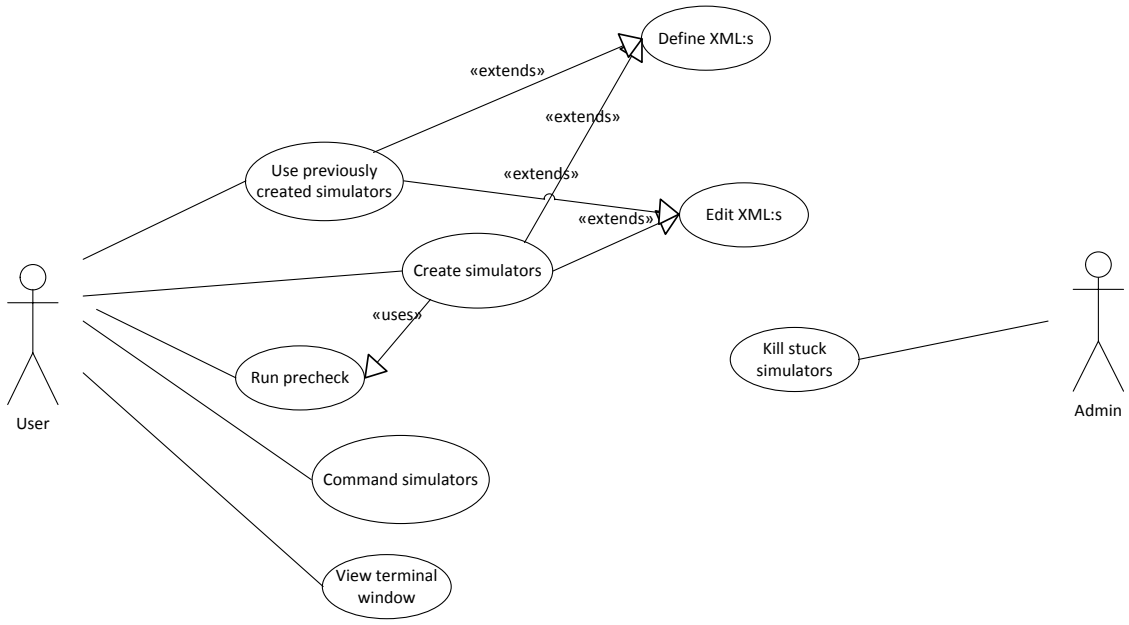
Kuvio 3 esittää sivua missä näytetään luodut simulaattorit sekä mahdolliset komennot joita voidaan antaa simulaattoreille. Tarkoitus on, että käyttäjä pystyy Napet GUI:ssa tekemään samoja toimintoja kuin Napetissa mutta graafisesti.



Kuvio 2: Suunniteltu aloitussivu



Kuvio 3: Suunniteltu simulaattorisivu



Kuvio 4: Käyttötapauskaavio

5.1.2 Viikko 2

Ensimmäinen versio ohjelmasta oli tehty mahdollisimman yksinkertaiseksi. Tarkoitus oli ainoastaan saada ohjelma ottamaan yhteys ssh:n ylitse, antaa komento joka käynnistää masterCLI:n ja tulostaa masterCLI:n antaman syötteen. Kuviossa 5 näemme aloitussivun, johon käyttäjä antaa kirjautumistiedot ja kuviossa 6 ohjelma on käynnistänyt masterCLI:n ja näyttää terminaali-ikkunan tekstin. Tässä vaiheessa masterCLI:tä ei pysty vielä hallitsemaan. MasterCLI:n vaatimia xml-tiedostoja ei myöskään anneta, vaan masterCLI käyttää oletustiedostoja.

Kuvio 5: Ensimmäinen versio aloitussivusta

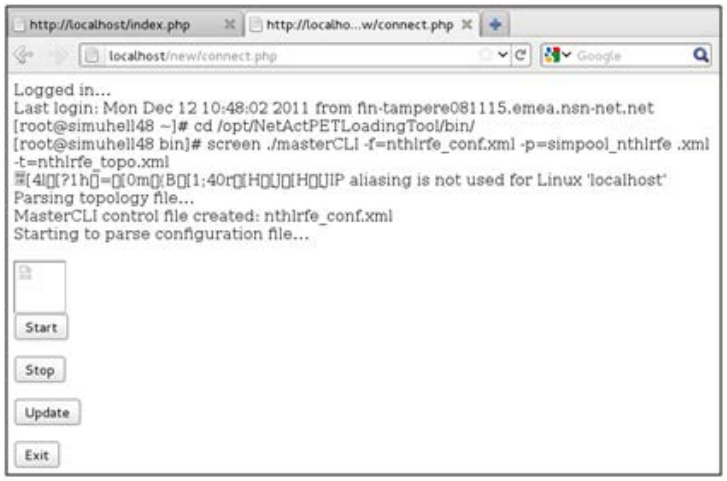
Kuvio 6: Ohjelma tulostaa vielä tässä vaiheessa ainoastaan ssh:n välityksellä saadun tekstin

5.1.3 Viikko 3

Ohjelmaan on nyt lisätty mahdollisuus hallita masterCLI:tä. Simulaattoreille on lisättynä start-, stop-, update- ja exit-nappulat (kuvio 7). Start-nappula käynnistää kaikki simulaattorit; toistaiseksi simulaattoreita ei pysty käynnistämään erikseen.

MasterCLI käynnistetään nyt screen:in sisälle, jolloin masterCLI ei sammu kun komennot olivat suoritettu ja yhteys katkeaa. Ennen jokaista komentoa otetaan uusi yhteys ssh:n ylitse ja avataan screen jossa Napet on käynnissä.

Etusivulle meneminen on nyt mahdotonta, jos masterCLI on käynnissä. Tämä estää käyttäjää käynnistämästä useamman masterCLI:n.



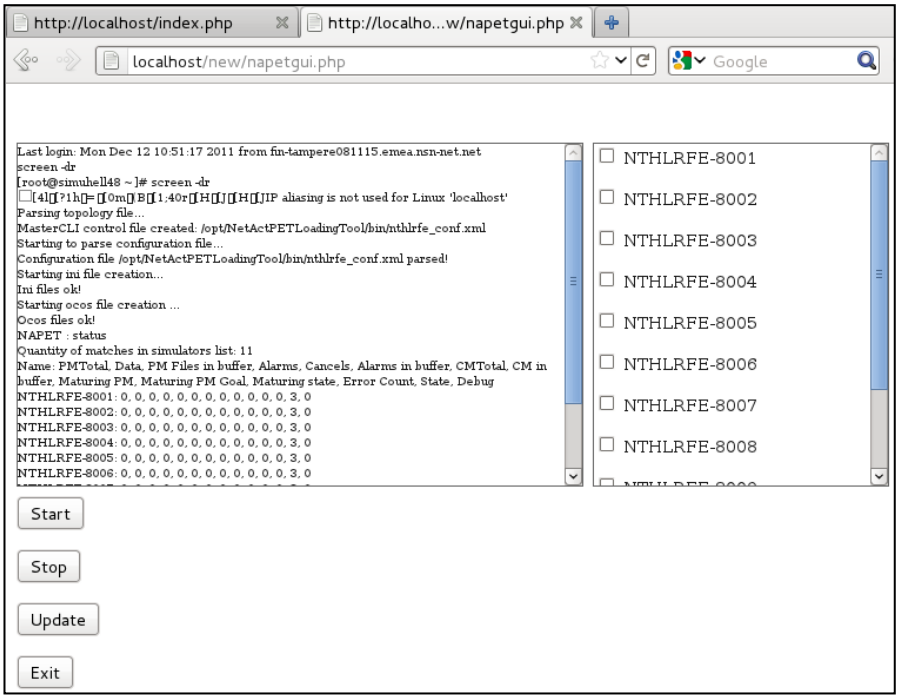
Kuvio 7: Simulaattorinäkymä

5.1.4 Viikko 4

Melko nopeasti päätimme, että sisäänkirjautuminen ei ollut oleellinen osa ohjelmaa, joten se ominaisuus päätettiin poistaa. Ohjelman lähdekoodiin on laitettu kirjautumistiedot, jolloin sisäänkirjautuminen hoidetaan käyttäjän tietämättä asiasta. Etusivu (kuvio 8) vaatii nyt käyttäjältä xml-tiedostot, jotka annetaan masterCLI:lle parametrina. Näiden tiedostojen avulla luodaan simulaattorit.

Kuviossa 9 näemme miten simulaattorinäkymä on muuttunut. Vasemmalla on terminaali-ikkunasta saatu teksti ja oikealla on listattuna simulaattorit. Simulaattoreita ei pysty edelleenkään käynnistämään erikseen.

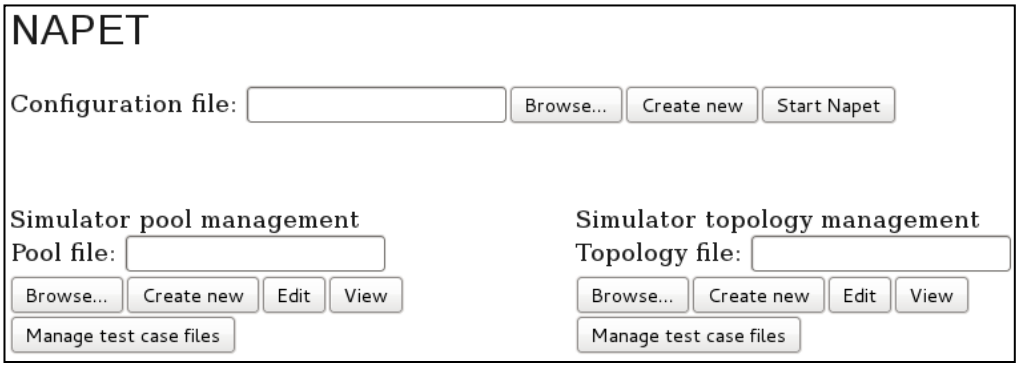
Kuvio 8: Sisäänkirjautuminen on poistettu etusivulta



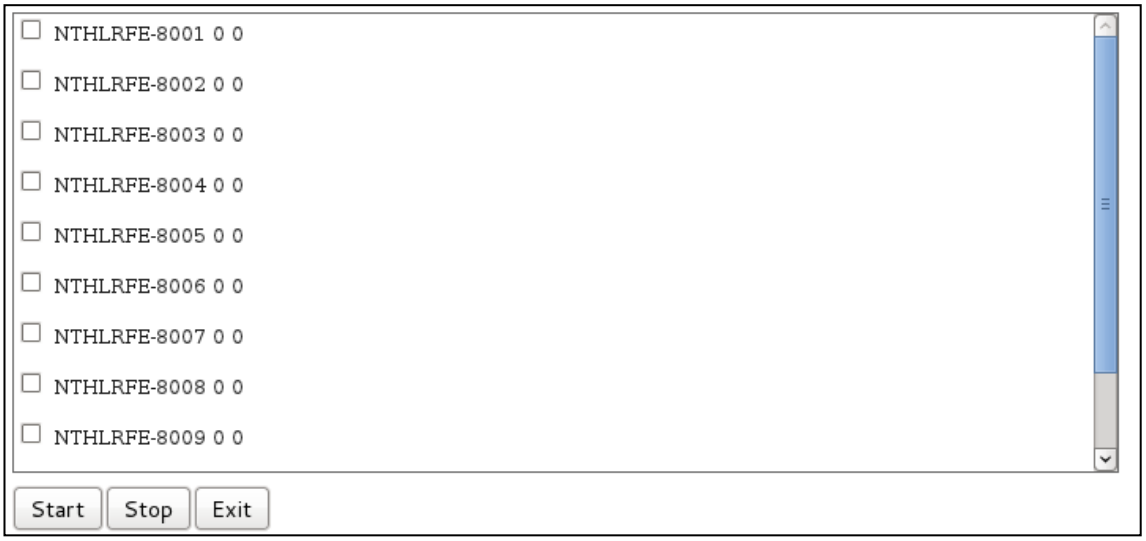
Kuvio 9: Simulaattorit ja terminaaliteksti omissa ruudukoissa

5.1.5 Viikko 5

Etusivu (kuvio 10) muokattiin vastaamaan paremmin ohjaajan visiota(kuvio 2). Nappulat eivät toimi, paitsi ”Start Napet”. Tarkoitus on ajan myötä lisätä napeille toimintoja. Simulaattorinäkymässä on konsoli-ikkuna poistettu, sen sijaan simulaattori-ikkuna on levennetty ja nappulat laitettu peräkkäin (kuvio 11).



Kuvio 10: Etusivun ulkomuotoa on muokattu



Kuvio 11: Terminaaliteksti on poistettu kokonaan

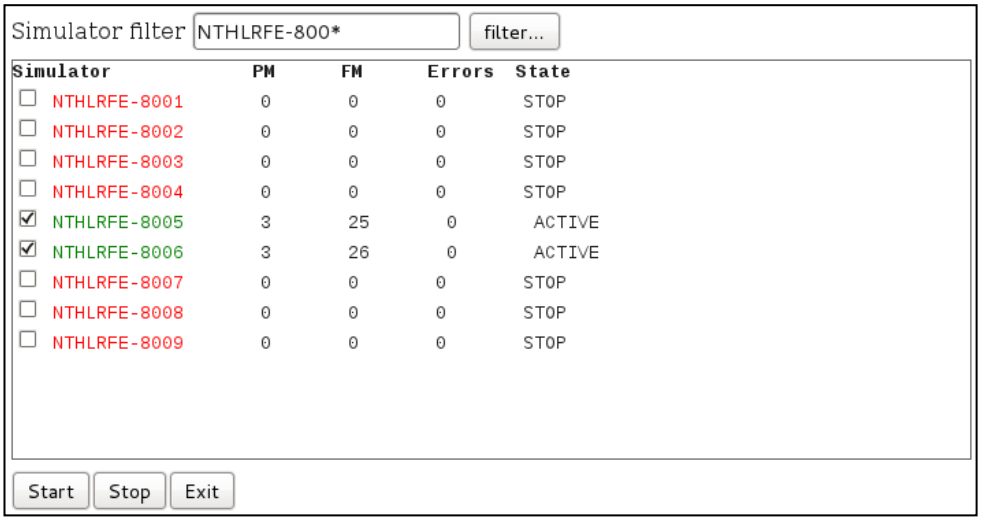
5.1.6 Viikko 6

Simulaattorinäkymään lisätty mahdollisuus filtteröidä näkyvissä olevia simulaattoreita, esim. hakuteksti ”NTHLRFE-800*” näyttää ainoastaan simulaattorit 1 – 9. Simulaattori ” NTHLRFE-8010” ei näytetä, koska se ei täsmää haun kanssa (kuvio 12).

Simulaattoreita pystyy nyt erikseen käynnistämään ja pysäyttämään. Simulaattori-ikkunassa simulaattorit näkyvät värikoodeilla, jotka kertovat niiden tilan.

Simulaattori-ikkunassa on nyt mittaustietoa, joita testaajat tarvitsevat työssään.

- PM eli mittausarvo
- FM eli hälytysarvo, ilmestyy kun simulaattorissa tapahtunut hälytys
- Errors eli virhetilanne, ilmestyy kun simulaattorissa tapahtunut ei toivottua
- State eli simulaattorin tila, kertoo onko simulaattori pois päältä, pysäytetty, tyhjäkäynnillä tai käynnissä.

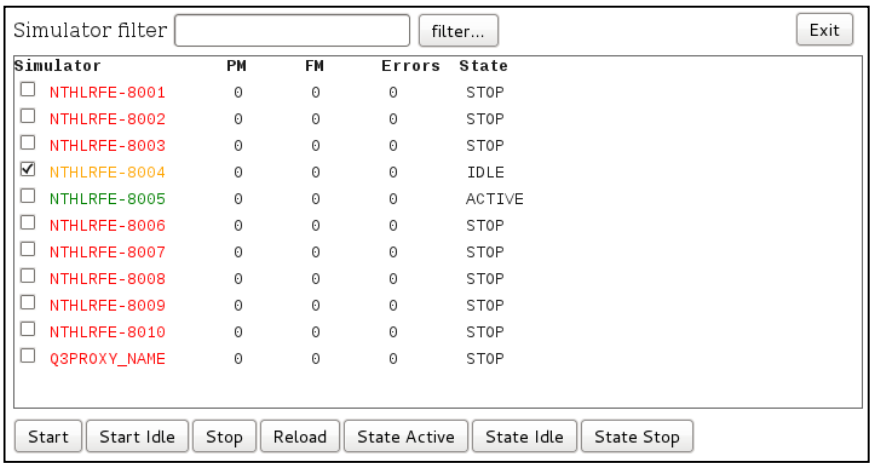


Kuvio 12: Filter-toiminto, mittausdataa ja värejä on lisätty

5.1.7 Viikko 7

Simulaattoreita pystyy nyt hallitsemaan paremmin. Käyttäjä pystyy käynnistämään simulaattorit tyhjäkäyntiin, jolloin simulaattorit ovat käynnissä, mutta eivät tuota minkäänlaista dataa.

”State” komennot on lisätty. Nämä mahdollistavat esim. käynnissä olevan simulaattorin tilan muuttamisen tyhjäkäynniksi. Jos simulaattori on käynnissä ja käyttäjä yrittää käynnistää saman simulaattorin tyhjäkäynnille painamalla ”Start Idle” nappia, ei masterCLI suostu käynnistämään simulaattoria uudestaan ja simulaattorin tila pysyy muuttumattomana. Tätä varten on State-napit, jolloin simulaattoria ei tarvitse käynnistää uudestaan ja mittausarvot säilyvät.



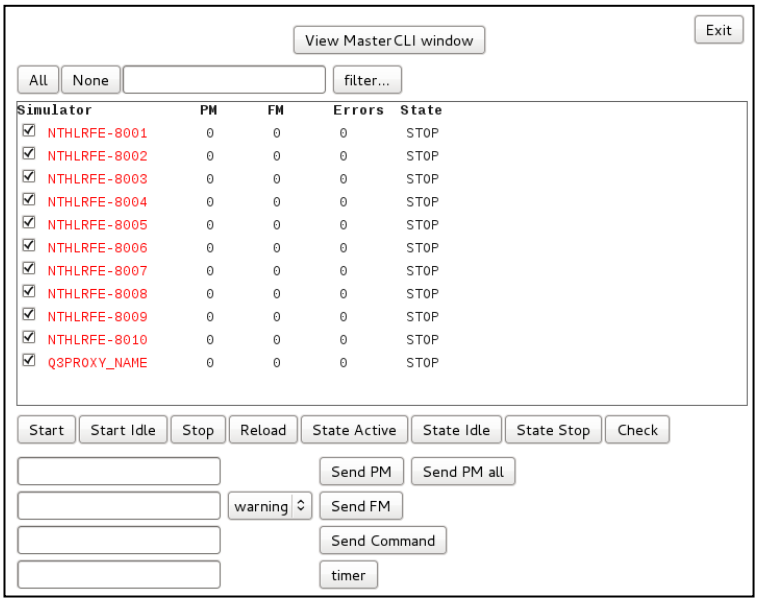
Kuvio 13: Komentoja ja Exit-nappi on lisätty

5.1.8 Viikko 8

Napit All ja None lisättiin etusivulle (kuvio 14). Näiden avulla voi valita joko kaikki simulaattorit tai ei yhtään. Tämä auttaa käyttäjiä, joilla on useita kymmeniä, joskus myös satoja, simulaattoreita valitsemaan haluamansa simulaattorit. Tätä ominaisuutta olivat käyttäjät toivoneet työn helpottamiseksi.

Käyttäjä pystyy nyt lähettämään simulaattoreille PM- ja FM-arvoja. Send Command-napin avulla käyttäjä voi antaa komentoja suoraan masterCLI:lle. Timer määrittää kuinka monen minuutin välein simulaattoritiedot päivitetään.

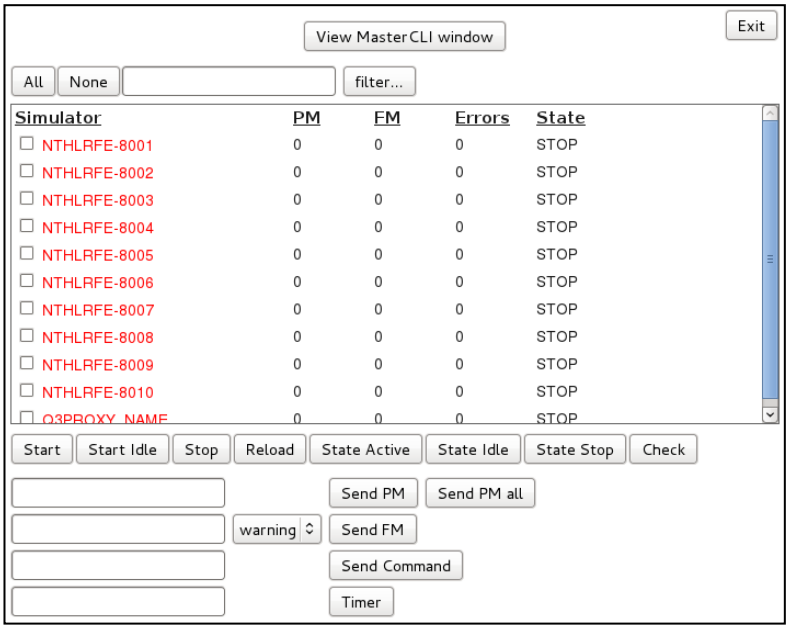
View MasterCLI window-nappi on lisätty mutta toiminnallisuus puuttuu. Tarkoitus on, että nappi avaa uuden ikkunan mistä käyttäjä pystyy seuraamaan konsolin tapahtumia.



Kuvio 14: Simulaattorinäkymä

5.1.9 Viikko 9

Simulaattorin data on nyt laitettu taulukkoon. Reunoja ei näy, mutta data pysyy siististi omissa taulukoissa eikä ulkomuoto rikkoonnu vaikka dataa tulee enemmänkin. Esimerkiksi kuviossa 12 näemme miten epäsiististi data sijoittuu kun taulukkoa ei ole vielä implementoitu.



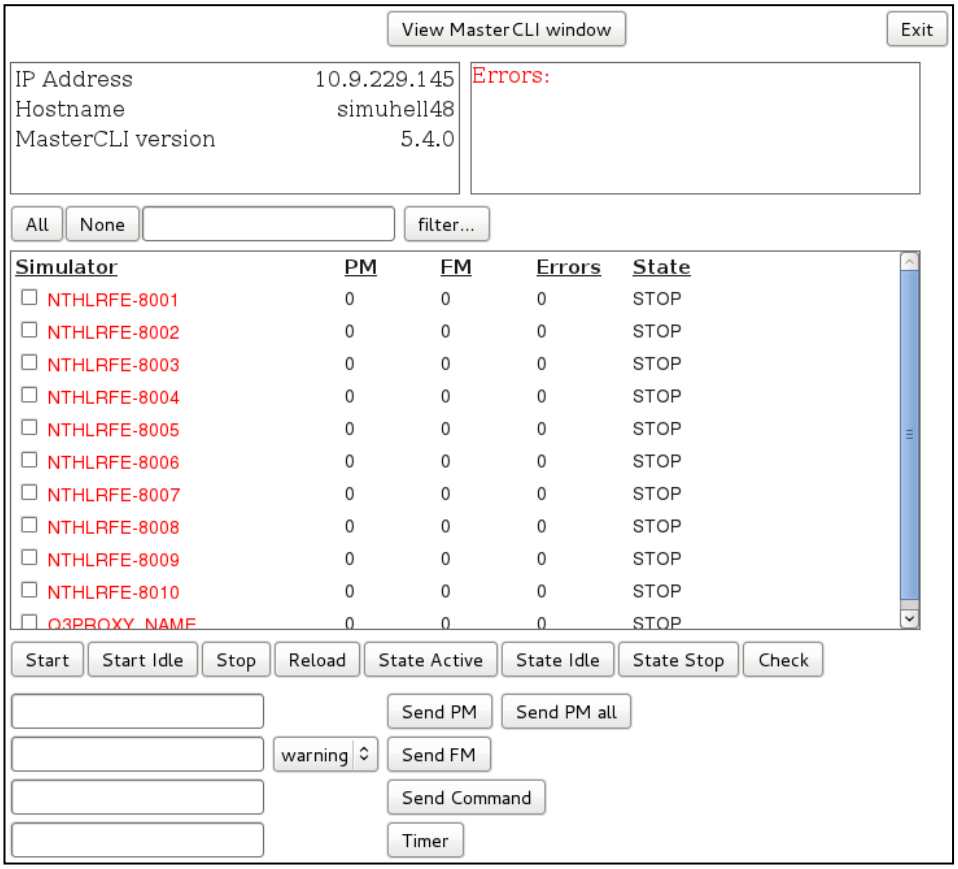
Kuvio 15: Simulaattorit ja mittausdata on laitettu taulukon sisälle

5.1.10 Viikko 10

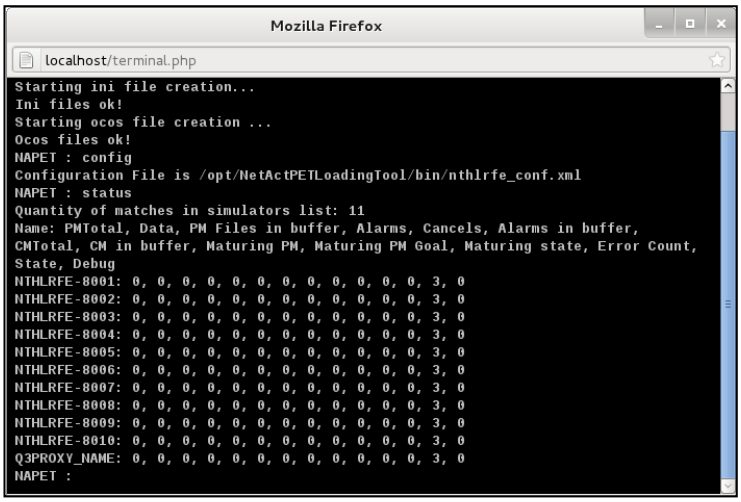
Infolaatikko kertoo tärkeimmät tiedot palvelimesta jossa masterCLI pyörii. Tämä ominaisuus lisättiin, kun käyttäjät halusivat nähdä paremmin tietoa palvelimesta. Infotaulu näkyy vasemmassa yläkulmassa kuviossa 16.

Virhelaatikko tulostaa virheilmoitukset niiden ilmaantuessa. Kaikki tekstit, jotka tulostetaan ulos stderr–muodossa, tulostuu tähän laatikkoon. Virheet tallennetaan tiedostoon, jota luetaan aina sivua ladattaessa.

View MasterCLI window–nappi on nyt toiminnassa. Nappia painamalla avautuu uusi ikkuna, johon tulostuu terminaalin teksti (kuvio 17). Teksti päivittyy 5 sekunnin välein. Tämä ominaisuus ei ole tarkoitettu peruskäyttäjille vaan niille, jotka haluavat nähdä mitä terminaalissa täsmälleen tapahtuu esim. virhetilanteessa.



Kuvio 16: Lisätty info- ja virhelaatikko



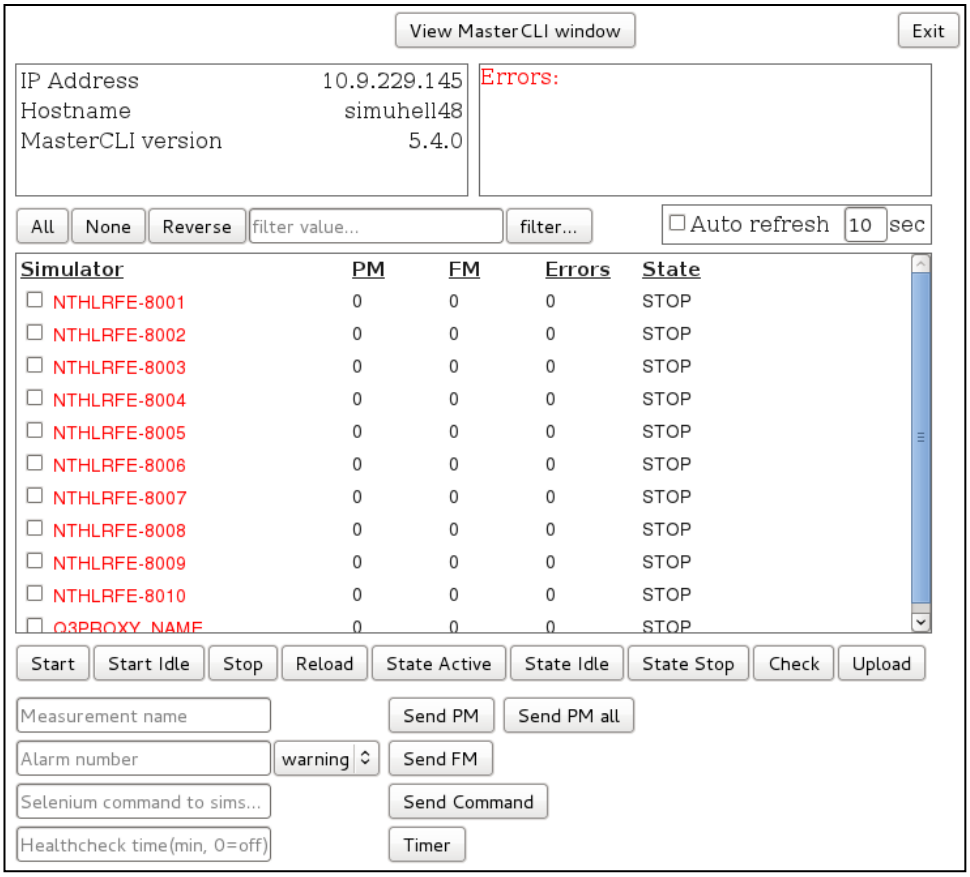
Kuvio 17: Terminaali

5.1.11 Viikko 11

Ohjelman tärkeimmät toiminnallisuudet ovat tässä vaiheessa luotu, jonka ansiosta tällä viikolla luotiin pieniä funktioita, jotka tekivät ohjelmasta käyttäjäystävällisempää.

Simulaattoritietoja ei päivitetä automaattisesti ja tämä heikentäisi testaajien tehokkuutta. Ohjelmaan lisättiin ominaisuus joka päivittää simulaattoritietoja tietyn ajan välein. Tämä toteutettiin JavaScript-funktiolla, joka välittömästi tunnistaa jos toiminto on käynnistetty ja päivittää tiedot välittömästä. Tämän jälkeen tiedot päivitetään annettujen parametrien välein.

Käyttäjät olivat myös toivoneet tekstilaatikkoihin selitykset, ohjelman selkeyttämiseksi. Kuvio 18 näemme uudet tekstilaatikot selityksineen.



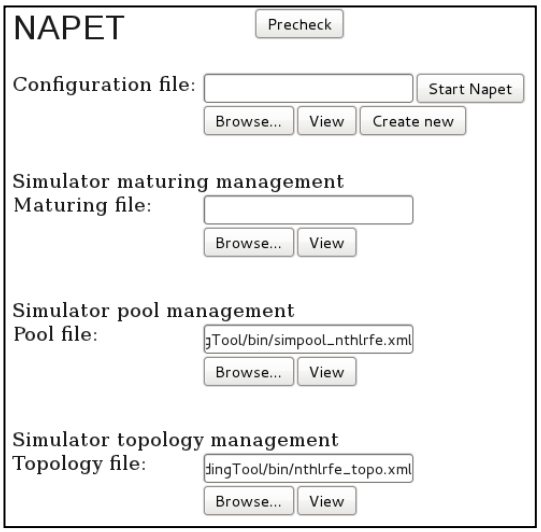
Kuvio 18: Sivun automaattinen päivitys ja laatikoiden selitykset lisätty

5.1.12 Viikko 14

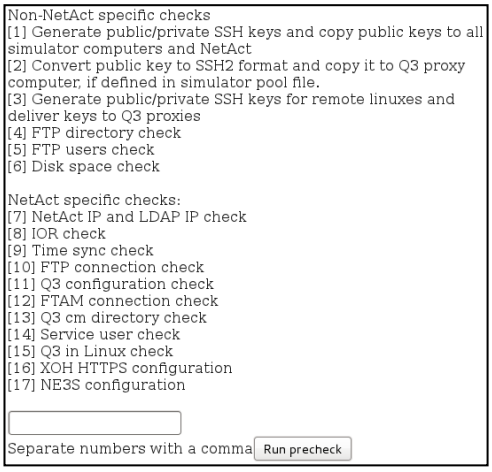
Kun simulaattorinäkymä alkoi olla valmis, oli aika keskittyä etusivuun. Kuviossa 2 esitetty suunnitelma ei ollut toteutuksen kannalta järkevä, joten jouduin muokkaamaan ulkonäköä. Etusivulla pystyy aluksi muokkaamaan ainoastaan konfiguraatio ja maturing tiedostoa (kuvio 19). Painamalla ”create new” –nappia tulevat muut vaihtoehdot esille (kuvio 20).

Ennen kuin simulaattoreita pystyy käynnistämään, on suositeltavaa ajaa precheck-skripti, joka tarkistaa että olosuhteet ovat optimoitu ja korjaa mahdolliset puutteet. Etusivulle on lisättynä precheck-nappi (kuvio 19), jota painamalla avautuu uusi ikkuna (kuvio 21) mistä käyttäjä voi valita ajettavat tarkistukset. Käyttäjä voi myös halutessaan painaa etusivulta ”Run precheck now” –nappia (kuvio 19), joka suorittaa automaattisesti kaikki tarkistukset(kuvio 22).

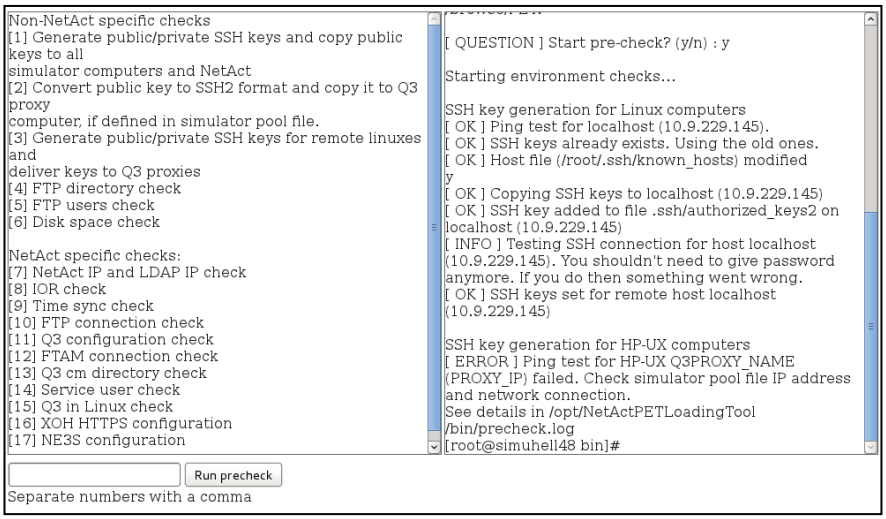
Kuvio 19: Uusi etusivu



Kuvio 20: Pool- ja topology-kohdat ovat alun perin piilossa



Kuvio 21: Precheck toiminnassa



Kuvio 22: Precheck näyttää skriptin syötteen tarkistuksien jälkeen

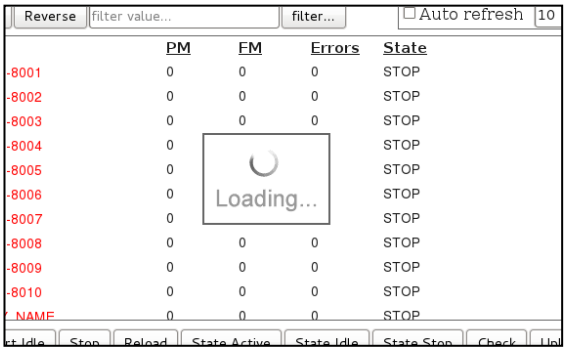
5.1.13 Viikko 17

Ainoa muutos tälle viikolle oli varmistus, joka kysytään painettaessa Exit–nappia (kuvio 23). Tämä toive tuli testikäyttäjältä, joka oli vahingossa sulkenut ohjelman ja samalla kaikki simulaattorit. Sama testikäyttäjä oli myös toivonut latausanimaatiota, jotta käyttäjä tietää milloin ohjelma suorittaa jotain komentoa (kuvio 24). Animaatio toteutettiin gif-tiedostolla joka piilotetaan heti sivun latauduttua. Kun käyttäjä painaa esim. Start-komentoa, laitetaan kuva näkyviin. Kun komento on suoritettu, latautuu sivu uudestaan ja animaatio menee oletuksena jälleen piiloon.

Suurin työ tällä viikolla oli kuitenkin käyttöohjeen suunnittelu ja toteutus (liite 1). Tämän lisäksi tehtiin asennusohjetta rhel 4-ympäristöön(liite 2). Asennusta varten tehtiin myös paljon tutkimusta, sillä asiaa hankaloitti rhel 4 tuen puute. Lue tarkemmin kappaleesta 6.

Osaamisen siirto tehtiin Tiedon Puolan yksikköön. Ohjelman valmistuttua he ovat vastuussa ohjelman kehityksestä ja virheiden korjaamisesta.

Kuvio 23: Käyttäjältä varmistetaan ohjelman sulkeminen

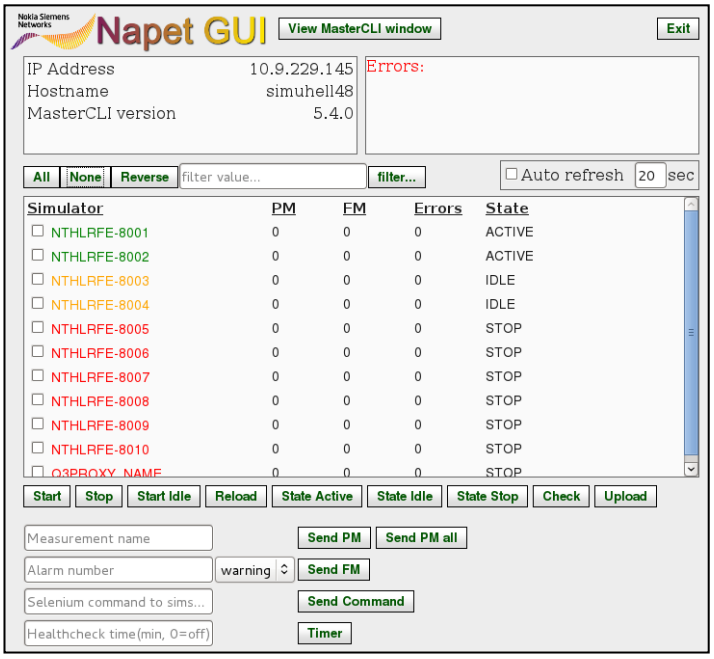


Kuvio 24: Latausanimaatio näkyy käyttäjälle komennon käsittelyajan aikana

5.1.14 Viikko 18

Ohjelma on nyt valmis! Tämän jälkeen mahdollisia muutoksia tekee ainoastaan Tieto. Toiminnallisia muutoksia ohjelmaan ei tullut, ainoastaan miellyttävämpi ulkonäkö. Käyttöjärjestelmäriippuvaiset napit ovat korvattu omilla napeilla, jotka ovat samat kaikissa käyttöjärjestelmissä. Ohjelman logo suunniteltiin Nokia Siemens Networksin värien mukaan ja lisättiin ohjelmaan. Ohjelman värimaailmaa muokattiin hieman tummemmaksi.

Kuvio 25: Etusivun lopullinen ulkonäkö



Kuvio 26: Simulaattorinäkymän lopullinen ulkonäkö

6 OHJELMAN ASENNUS RHEL 4-YMPÄRISTÖÖN

Kun ohjelmaa alussa suunniteltiin, oli tarkoitus asentaa PHP-palvelin, johon saisin itse päättää käyttöjärjestelmän sekä siihen tarvittavat paketit, ja yhtiön palveluksessa olevat ammattilaiset asentaisivat palvelimen käyttöliittymää varten. Kun ohjelma alkoi valmistua, päätimme että PHP-palvelin tulisi pyöriä samassa koneessa, jossa käyttöliittymän hallitsema Napet ohjelma pyörii. Napet toimii RHEL 4 käyttöjärjestelmässä.

6.1 RHEL 4 – Käyttöjärjestelmä palvelimille

RHEL (Red Hat Enterprise Linux) on Red Hatin kehittämä Linux-pohjainen käyttöjärjestelmä palvelimille. RHEL on laajasti käytettynä useissa eri yhtiössä.

RHEL 4 tuki lopetettiin vuonna 2009 ja tuotteen virallinen elinikä päättyy karkauspäivänä 2012. Uusin versio, RHEL 6, julkaistiin vuoden 2010 lopussa. (Red Hat: Red Hat Enterprise Linux Server 2012).

6.2 PHP-palvelimen asennus

Suurin ongelma oli, että RHEL 4:n tuki lopetettiin 2009, joten uusimpia paketteja, kuten esim. PHP5, tukea ei löytynyt. Apache-palvelin oli jo valmiiksi asennettuna, mutta PHP oli versio 4, ja käyttöliittymä vaati version 5. Latasin PHP:n verkkosivuilta uusimman vakaan version lähdekoodista ja käänsin sen itse RHEL 4:lle. Vaikeuksia syntyi siinä, että PHP5 asennuksessa oli riippuvuuksia, joita Yum ei kyennyt automaattisesti asentamaan, koska RHEL 4 tuki on lopetettu. Löydettyäni ja asennettuani nämä paketit Internetistä RHEL 4:lle, sain viimeinkin PHP5-paketin asennettua.

6.3 PHP:n ssh tuki

Seuraavana haasteena oli PHP:n ssh-tuki. Tämä tuki ei tule automaattisesti PHP5:n kanssa, vaan se tulee asentaa erikseen. Jälleen kerran, suoraa tukea RHEL 4:lle ei löytynyt. Tällä kertaa en joutunut itse kääntämään pakettia lähdekoodista, vaan löysin valmiin rpm-paketin, josta pystyin asentamaan tuen. Ongelmana olivat jälleen kerran

paketin riippuvuudet, jotka ratkaisin asentamalla jokaisen paketin, etsimällä ne ensin Internetistä. Tämän jälkeen asennus onnistui ilman ongelmia.

6.4 Ohjelman siirtäminen RHEL 4-ympäristöön

Kun PHP- ja Apache-palvelin saatiin toimimaan RHEL 4-ympäristössä, oli tehtävänä siirtää ohjelma palvelimelle. Tämä tuotti suuria ongelmia, joista ensimmäinen oli PHP:n ssh-paketti, joka oli vanha versio. Koska uusia versioita ei ollut saatavana rhel 4:lle, piti Napet GUI:ta muokata käyttämään ssh-pakettia paremmin.

6.4.1 Ongelmien korjaaminen

Ohjelman yrittäessään lähettää ssh–komento PHP:stä, tulivat noin puolet komennoista perille. Syy selvisi usean päivän tutkimisen jälkeen, että komento itse asiassa tuli perille ja toteutettiin, mutta PHP:n ssh-paketit eivät aina osanneet ottaa komennon jälkeistä syötettä vastaan. Koko Napet GUI:n toiminnallisuus perustuu syötteen tulkitsemiseen ja kun syötettä ei koskaan saatu ei Napet GUI kyennyt päättämään jatkotoimista.

Ongelma saatiin ratkaistua antamalla lukukäskyn uudestaan hetken päästä niin kauan kunnes vastaus saadaan. Tämän muutoksen jälkeen ei samanlaisia ongelmia enää ilmennyt.

7 TESTAUS

Testaus on tärkeä osa ohjelmistokehitystä. Testauksen tarkoituksena on löytää ohjelmointivirheitä, mutta samalla myös selviää, jos ohjelmisto täyttää sille asetetut vaatimukset ja että ohjelmaa pystyy normaali käyttäjä käyttämään. (Haikala & Märijärvi 2006, 283 - 287.)

7.1 Erilaiset testaustyypit

Konkreettisesti testaus on erilaisten prosessien ajamista ohjelmalle. Kuitenkin näitä prosesseja eli tapoja testata on lukuisia. Koska testausprosesseja on lukuisia ja useat niistä lähes joka kerta tarpeellisia, on testaajan tietotaitojen oltava myös laajat. Todellisuus on kuitenkin useimmiten toista: testaaja hallitsee tietyn tyyppiset testaukset ja pyrkii näin ollen ajamaan ja toteuttamaan niitä. (Sorvo, 2010, 15.)

Testauksen voi jakaa kahteen eri alueeseen: toiminnallinen ja ei-toiminnallinen testaus. Toiminnallinen testaus sisältää toiminnallisuustestauksen, yhtäaikaisuustestauksen, asennustestauksen, aloitustestauksen ja konfiguraatiotestauksen. Ei-toiminnallinen testaus sisältää kuormitustestauksen, rasitustestauksen, tietoturvatestauksen ja käytettävyystestauksen. (Sorvo, 2010, 15 – 18).

7.2 Napet GUI:n testaus

Napet GUI:n testaus aloitettiin noin kuukausi ennen työsuhteen päättymistä. Testaajana oli kaksi tulevaa käyttäjää. Testaajat ovat normaalisti käyttäneet Napetia ilman graafista käyttöliittymää, joten he osasivat jo valmiiksi käyttää terminaaliohjelmaa. Kohdatessaan vaikeuksia, testaajat lähettivät virheilmoituksen tai kehitysehdotuksensa minulle sähköpostitse.

Testauksessa ei käytetty minkäänlaista automaatiota, koska se todettiin tarpeettomaksi. Testikäyttäjät eivät löytäneet mitään suurempia virheitä. Enimmäkseen he antoivat kehitysehdotuksia ulkonäöstä ja toivoivat että ohjelma toteutettaisi komennot nopeammin. Jälkimmäinen ei ollut mahdollinen, sillä Napet GUI toteutti kaikki komennot välittömästi, jonka jälkeen ohjelma jäi odottamaan, että terminaaliohjelma on

toteuttanut annetut komennot. Kun komennot on toteutettu, jatkoi Napet GUI taas normaalisti.

Osa virheistä ohjelmassa johtui PHP:n ssh-paketeista. Nämä paketit olivat vanhoja ja viallisia, johtuen RHEL 4:sta. Tämä tuotti testaajille välillä päänvaivaa. Tämä ongelma korjattiin lisäämällä funktioita, jotka yhteysongelmien sattuessa yrittivät korjata vian ja yhdistivät uudestaan. Jos komennon lähetyksen jälkeen vastausta ei saatu, kyseltiin vastausta toistuvasti. Saatu vastaus oli myös joskus vain osittainen. Tämä korjattiin kysymällä vastausta uudestaan. Tällöin saatiin joko puuttuva osa tekstistä tai koko teksti alusta loppuun. Tämä uusi tekstinpätkä lisättiin edellisen loppuun, jolloin saatiin koko vastausteksti. Osa ratkaisuista oli valitettavan monimutkaisia, mutta välttämättömiä toiminnan kannalta. Ongelma esiintyi kuitenkin erittäin harvoin, joka myös poistui lopullisesti uusien funktioiden avulla. Vaihtoehto olisi käyttää jotain muuta Linux-käyttöjärjestelmää, mutta se ei ollut mahdollista tässä projektissa.

Osa testaajien toivomista ulkonäkömuutoksista toteutettiin lopulliseen versioon. Kuvio 25 ja kuvio 26 näemme miten ohjelman tausta, nappuloiden ulkomuoto yms. on muutettu. Suurin osa näistä on testaajien toiveiden mukaiset.

8 VERSIONHALLINTA

Versionhallintaa käytetään ohjelmistoprojekteissa pitämään kirjaa tiedostoihin tehdyistä muutoksista ja vanhemmista versioista. Hyviin toimintatapoihin kuuluu lähdekoodin pitämistä versiohallinnassa. (Better Explained: A visual guide to version control 2012.)

8.1 Subversion

Subversion (Svn) on käytetyin versionhallintajärjestelmä. Toiminta pohjautuu keskuspalvelimeen, johon kaikki muutokset ja versiot tallennetaan. Svn on avoimeen lähdekoodiin perustuva projekti, joka aloitettiin vuonna 2000. Svn luotiin korvaamaan cvs, joka on vanhentunut versionhallinta. (Apache: Apache Subversion 2012.)

8.2 Mercurial

Mercurial on pythonilla kehitetty versionhallintajärjestelmä. Mercurial muistuttaa osittain subversionia, mutta pienillä lisäyksillä. Mercurialissa on mahdollisuus tehdä projektiin muutoksia, jotka menevät omaan yksityiseen versiohallintaan, jotka lisätään keskuspalvelimeen vasta kun käyttäjä niin haluaa.

Mercurial on tehty mahdollisimman yksinkertaiseksi, missä piilee myös sen heikkous. Mercurial ei sisällä kehittyneempiä komentoja, joita löytyy esimerkiksi Gitistä. (Mercurial: Mercurial Guide 2012.)

8.3 Git

Git on alun perin Linus Torvaldsin kehittämä versiohallinta Linux kernelin kehittämistä varten. Git:in sanotaan olevan vaikeampi oppia kuin Mercurial mutta Git on tehokkaampi ja sopii paremmin isojen projektien hallintaan, joihin tulee jatkuvasti muutoksia. (Wikivs: Git vs. Mercurial 2012.)

8.4 Napet GUI:n versiohallinta

Valitsin versiohallintaa varten Subversionin. Tämä johtui siitä että työskentelin Linuxissa ja Svn toimii moitteettomasti siinä. Myös Mercurial ja Git olisivat tähän

käyneet, mutta koska en tarvinnut Gitin ominaisuuksia, ja Mercurial toimii parhaiten graafisella käyttöliittymällä, päädyin Subversionin.

Keskuspalvelimena käytin samaa palvelinta, jota käytin verkkopalvelimena. Tämä riitti minulle, koska olin ainoa kehittäjä ja en tarvinnut tehokasta keskuspalvelinta versionhallintaa varten.

Lisäsin versionhallintaan uuden version aina kun olin saanut jonkin osan lisättyä ohjelmaan. Tämä mahdollisti minun palata aikaisempaan versioon, jos uusi versio osoittautui vialliseksi. Ilman versionhallintaa olisi työmäärä virheiden sattuessa varmasti moninkertaistunut. Tällöin olisin joutunut käsin poistamaan kaikki muutokset ja sen jälkeen vasta etsimään mikä muutos rikkoi ohjelman.

9 LOPULLISEN VERSION JULKAISEMINEN

Lopullinen versio julkaistiin käyttäjille samana päivänä kun työsuhteeni loppui. Itse olin ehdottanut, että tämä tehdään viimeistään kaksi viikkoa aikaisemmin. Viimeiset viikot olisivat käyttäjät saaneet testata ohjelmaa ja olisin ehtinyt korjata mahdolliset virheet. Tämä ei valitettavasti toteutunut minusta riippumattomista syistä. Koska en ole enää Nokia Siemens Networksin palkkalistoilla, en ole saanut mitään tietoa ohjelman toiminnasta tai siitä, onko lopullinen versio käyttäjien mieleen.

Ohjelmaa ylläpitää Tieto Oy, Puolan yksikkö. Vaikka nykyinen ylläpitäjä on ammattilainen, on ylläpito varmaan haasteellista, kun kyseessä on toisen henkilön tekemä ohjelma. Ongelmatilanteessa minä olin ainoa henkilö, joka osasi neuvoa ja korjata ongelmia. Nyt kun lopullinen versio julkaistiin, kääntyvät käyttäjät nykyiseen ylläpitäjään, joka ei ole ohjelmoinut riviäkään ohjelmasta. Tämä tuottaa ongelmia, jotka olisi vältetty, jos lopullinen versio olisi annettu käyttäjille kun työsuhteeni oli vielä voimassa.

10 LÄHTEET

Agile Finland. 2012. Ketterät menetelmät. [online]. Luettu 22.03.2012. http://agilefinland.com

Apache. 2012. Apache Subversion. [online]. Luettu 22.03.2012. http://subversion.apache.org/

Better Explained. 2012. A visual guide to version control. [online]. Luettu 15.03.2012. http://betterexplained.com/articles/a-visual-guide-to-version-control/

Haikala & Märijärvi 2006. Ohjelmistotuotanto. Helsinki: Talentum, 283 - 287

Java. 2012. Java Online. [online]. Luettu 23.03.2012. http://www.java.com/en/

The Javascript source. 2012. JavaScript. [online]. Luettu 25.03.2012 .http://www.javascriptsource.com/

Mercurial. 2012. Mercurial Guide. [online]. Luettu 22.03.2012. http://mercurial.selenic.com/guide/

Nokia Qt. 2012. Cross-platform application and UI framework. [online]. Luettu 21.03.2012. http://qt.nokia.com/

PHP. 2012. PHP Hypertext Preprocessor. [online]. Luettu 12.03.2012. http://php.net/

Red Hat. 2012. Red Hat Enterprise Linux Server. [online]. Luettu 02.03.2012 . http://www.redhat.com/products/enterprise-linux/server/

Scrum. 2012. Improving The Profession of Software Development. [online]. Luettu 21.03.2012. http://www.scrum.org/

Sorvo, M. 2010. Testauksen liittäminen ohjelmistoprojektiin. Tietojenkäsittelyn koulutusohjelma. Tampereen ammattikorkeakoulu. Opinnäytetyö. 15 - 18

TmForum. 2012. Conformance Certification Nokia Siemens Networks NetAct. [online] Luettu 26.4.2012. http://www.tmforum.org/NokiaSiemensNetworks/8815/home.HTML

W3Schools. 2012. HTML Intro. [online] Luettu 27.4.2012. http://www.w3schools.com/html/html_intro.asp

Wikivs. 2012. Git vs. Mercurial. [online]. Luettu 22.03.2012. http://www.wikivs.com/wiki/Git_vs_Mercurial

11 LIITTEET

Liite1: Napet GUI:n käyttöohje

1 Introduction

This manual assumes that Napet GUI has been installed and configured on a server. Napet GUI has been tested on the following browsers: Firefox, Chrome and Opera. Napet GUI does not support Internet Explorer!

Napet GUI works by giving commands to the masterCLI program running on the server. If you close the browser while simulators are running, this will not close the masterCLI program. To close masterCLI, use the “Exit” -button (chapter 3.6).

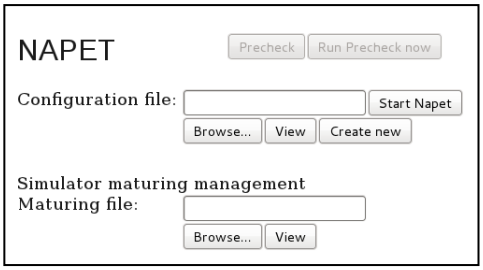
The pictures for this guide were taken on Fedora with Firefox. Some views might look different, depending on the operating system and browser being used.

Napet user guide:
https://isource.access.nokiasiemensnetworks.com/docman/view.php/1980/2138/NetAct_UsersGuide.doc

Wiki:
https://confluence.inside.nokiasiemensnetworks.com/display/OSSOPLSP/NetAct+PeT+Loading+Tool

2 Open Napet Gui

- Open the browser of your choice
- Type the ip-address of the server where Napet GUI was installed
- The main view should open up (picture 1)
  Note: If the main page was not opened, but instead the simulator view (chapter 2, picture 8), that means masterCLI was already running. To get to the main page, press “Exit” and “Ok” (warning: all simulators will be closed!)

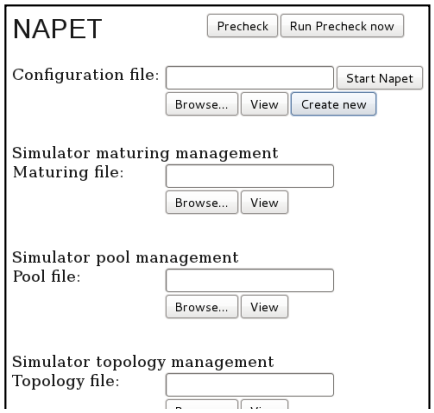


Picture 1

2.1 Running precheck

Precheck does environment checks for NetAct PeT Loading Tool environment before the simulation is started.

Precheck requires a pool file and a topology file in order to be run. If they are not specified, you will get an error message.



Picture 2

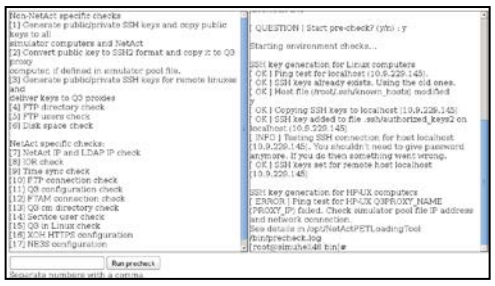
- Press the “Create new” button. You can now specify the pool and topology files (picture 2)
- At the “Simulator pool management”, press “Browse…” and select the pool file of your simulators. You can also just simply type the path of the pool file in the text box, eg. “/opt/NetActPETLoadingTool/bin/nthlrfe_conf.xml”.
  Note: The files have to be on the server where masterCLI is to be run.
- Specify the topology file as well

There are now two ways to run precheck. If you want to specify which check should be run, then press “Precheck”. If you want to run all the available check, press “Run Precheck now”.

2.1.1 Precheck

- Press the “Precheck” button, a new window will be opened (picture 3)
- From the text view, you can see all the checks that can be run. Type the checks in the text box that you want to run, e.g. “1,2,3”
- When you are ready to run the check(s), press “Run precheck”
- When the checks have been run, the output is printed for you (picture 4)
- When you are done with the checks, you can close the window

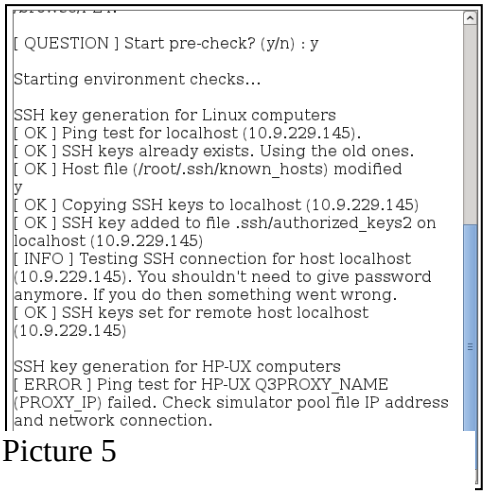
Picture 3



Picture 4

2.1.2 Run precheck now

- Press the “Run precheck now” button, a new window will be opened (picture 5), all checks will be run.
- The new window shows what was done during the checks. When you are done, you can close the window.



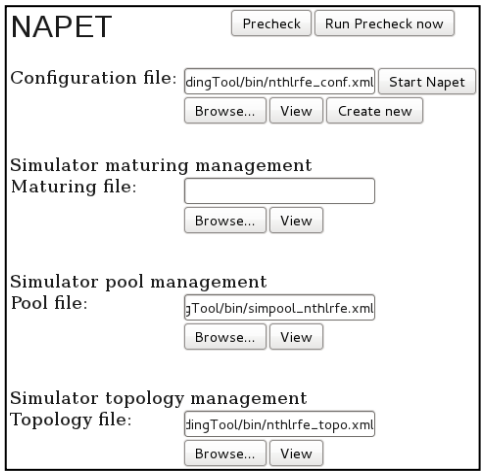
Picture 5

2.2 Start Napet GUI

If this is the first time running these simulators on this server, then pool and topology files must be defined. In this case, go to “2.2.1 Start Napet GUI with new simulators”. If you have run these simulators before, go to “2.2.2 Start Napet GUI with old simulators”.

2.2.1 Start Napet GUI with new simulators

- Press the “Create new” button to reveal the pool and topology management
- Define the configuration, pool, and topology file with the help of the “Browse…” buttons or by simply typing the path to the file (picture 6) Note: The files have to be on the server where masterCLI is to be run.
- If a file with the same name as the configuration file you specified exists, then it will be overwritten.
- Optionally you can specify a maturing file by adding the path to the file in the “Simulator maturing management” text box. If you do not know what this is, then leave it blank!
- Press the “Start Napet” button, go to part 3



Picture 6

2.2.2 Start Napet GUI with old simulators

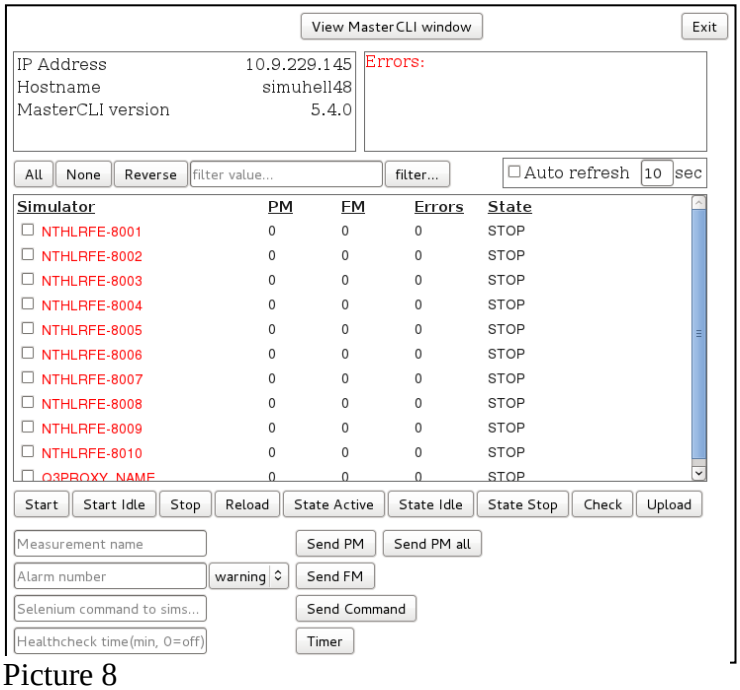
- Define the configuration file (picture 7)

- Optionally you can specify a maturing file by adding the path to the file in the “Simulator maturing management” text box. If you do not know what this is, then leave it blank!
- Press the “Start Napet” button, go to part 3

Picture 7

3 Napet GUI

In picture 8 you can see the GUI when Napet has been started. The simulator amount will differ, depending on the pool and topology files used when starting Napet GUI.



Picture 8

3.1 The info box

The info box is located in the upper left corner. It tells you the ip address, hostname and masterCLI version of the server.

3.2 The error box

The error box is located in the upper right corner. Whenever an error occurs, it will be displayed here. These errors are only those outputted by masterCLI. If an error occurs with Napet GUI, they will not be displayed here!

3.3 Simulator box

The simulator box is located in the middle of the screen (the biggest box). It has listed all the simulators that are available. It also tells the pm, fm, errors and states of the simulators.

To give commands to the simulators, select the simulators you want, and press the command button you wish to give to the selected simulators.

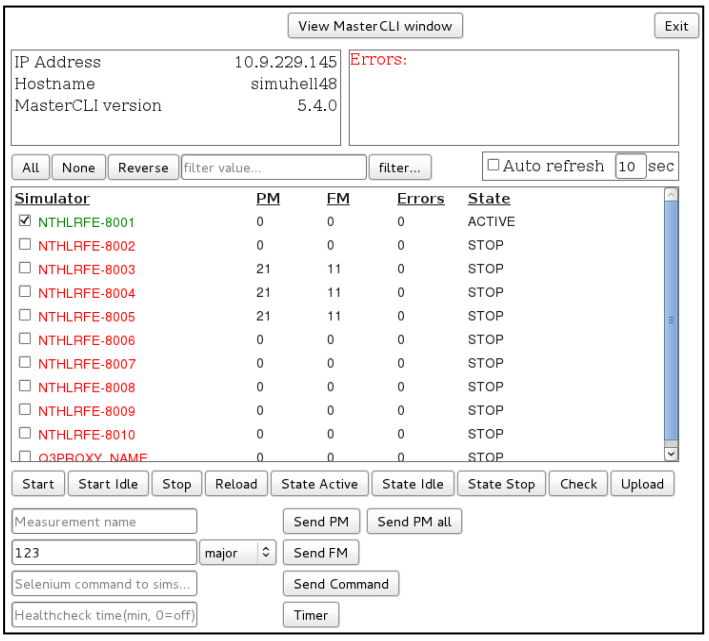
If you want to give the command to all the available simulators, you can either select all the simulators, or by not selecting any simulators and just pressing the command you want. When no simulators are selected, the command will be given to every simulator. Note: If you select every simulator and press the command you want, that will take a longer time than just pressing the command button without selecting any simulators, because Napet GUI will give the command separately to every simulator, instead of just giving the command once, which will apply to all available simulators.

3.3.1 Filter value

The filter value is located below the info box. It is for printing the simulators you wish to see. In picture 9 there are simulators NTHLRFE-8001 – 8010. If you only want to see simulators 8001-8009, do the following:

- Type in the filter value text box “NTHLRFE-800*”
- Press the “filter…” – button
- Notice that only simulators 8001 – 8009 are now shown

Any command now given to the simulators, will only affect those visible. Remove the text and press “filter…” to show all the simulators again.



Picture 9

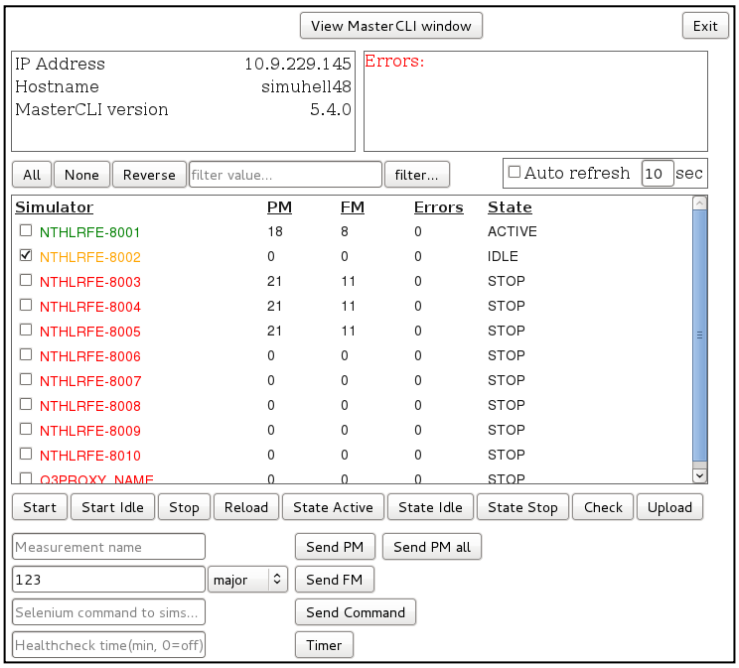
3.3.2 Starting a simulator

- To start simulators, first select the simulators you wish to start or select none to apply to all simulators
- Press the “Start” button, wait for the loading to complete
- Now the simulator should have started (picture 9). Notice that the simulator name has changed to green and that the state is now “ACTIVE”. This tells you that the simulator is running.

Note: Does not work on all simulators!

3.3.3 Starting a simulator in idle mode

- Select the simulators you wish to start, or select none to apply to all simulators
- Press the “Start Idle” button, wait for the loading to complete
- Now the simulators are started in idle mode. Notice that the simulator text has turned yellow and that the state is now “IDLE”. This tells you that the simulator is running but idle.



Picture 10

3.3.4 Stopping a simulator

- Select the simulator(s) you want to stop
- Press the “Stop” button, wait for load to finish
- The simulator is now stopped. Notice that the simulator(s) is now red and state changed to “STOP”.

3.3.5 Reload

Reloads the Test case of all (active and idle) NE3S simulators.

- Select the simulators you want to reload or none if you want to reload all
- Press the “Reload” button, wait for load to finish

3.3.6 State active, idle and stop

Changes the simulator state, e.g. from idle to active.

- Select the simulators which state you want to change or none if you want to change all the simulators state
- Press “State Active”, “State Idle” or “State Stop”, depending on what to what state you want to change to. Wait for load to finish
- The simulators have now changed states

Note: Does not work on all simulators!

3.3.7 Check

Gets the current status of the simulators (send health check msg).

- Select the simulators that you want to send the check command to or none if you want to send the check command to all available simulators
- Press the “Check” button, wait for load to finish

3.3.8 Upload

Upload ocos files into NAS.

- Select the simulators that you want to send the upload command to or none if you want to send the upload command to all available simulators
- Press the “Upload” button, wait for load to finish

3.3.9 Send PM

Command simulators to send a measurement.

- Select the simulators that you want to send the “send pm” command to or none if you want to send the “send pm” command to all available simulators
- Type the measurement name in the “Measurement name” text box
- Press the “Send PM” button, wait for load to finish
- You might need to press the “check” button to see any change in the simulator

3.3.10 Send PM all

Command simulators to send all test case measurements.

- Select the simulators that you want to send the “send pm all” command to or none if you want to send the “send pm all” command to all available simulators
- No need to type anything in the “Measurement name” box
- Press the “Send PM all” button, wait for load to finish
- You might need to press the “check” button to see any change in the simulator

3.3.11 Send FM

Command simulators to send an alarm (nbr=alarm number, severity =warning, minor, major, critical, cleared).

- Select the simulators that you want to send the “send fm” command to or none if you want to send the “send fm” command to all available simulators
- Type the alarm number in the “Alarm number” box
- Select the alarm type, warning, minor, major, critical or cleared
- Press the “Send FM” button, wait for load to finish

3.3.12 Send command

Command simulators to send Selenium command to all active simulators

- Type the selenium command to the “Selenium command to send” box
- Press the “Send Command” button, wait for load to finish

3.3.13 Timer

Change health check interval time (minutes 0=off).

- Type the value you want in to the “Health check time” box
- Press the “Timer” button, wait for load to finish

3.3.14 Auto refresh

Auto refresh is located between the simulator box and the error box. By default, the value is 20 seconds and the value can be from 20 to 999. If you type a value that is less than 20 seconds, it will automatically be changed back to 20 seconds.

Auto refresh, refreshes the page every 20 seconds (or whatever value is set) and automatically sends the “check” command, thus forcing the refresh of the simulators and fm, pm etc. values will be updated.

- Type the amount of seconds you want to be as the update interval
- Check the box next to “Auto refresh”, auto refresh is now on
- Uncheck the box to turn auto refresh off

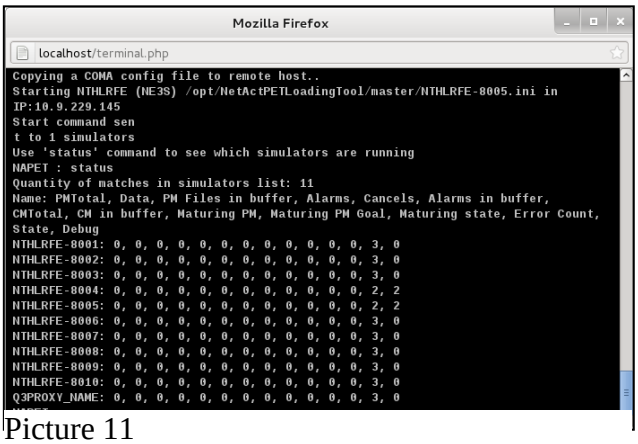
3.4 All, None and Reverse Buttons

These buttons are linked to the simulators check boxes.

- Press the “All” button to select all simulators
- Press the “None” button to unselect all simulators
- Press the “Reverse” button to unselect all currently selected simulators, and select all unselected simulators

3.5 View MasterCLI window

The “View MasterCLI window” button is located on the top of the window. It shows masterCLI:s terminal (picture 11). You can’t input anything to the terminal through this window, it is only



Picture 11

meant to quickly check if something went wrong in the terminal, or if a user just simply wants to see what goes on in the background.

The window is automatically updated every 5 seconds.

3.6 Exit

To exit Napet GUI, press the “Exit” button which is located in the upper right corner, all simulators will be stopped. It is suggested that you first stop all the simulators, then press “Exit”.

NOTE: Closing your browser will not stop the simulators.

Liite 2: Asennusopas

1 Introduction

This manual is meant for installing Napet gui on a rhel 4 server. On other Linux distros, go to

chapter 8.

Note: Root privileges are required!

2 Copy files to the rhel 4 server

Copy the following folders the server:

- php5
- php_ssh
- screen

3 Screen

Screen is needed to allow masterCLI to run on the server, when there is no active connections from the user. If there is an error saying package already installed, you may proceed.

- cd screen
- rpm -ivh screen-4.0.2-5.i386.rpm

4 PHP5

Rhel 4 already has php4 installed, but because php5 is available, it is highly recommended to upgrade. Napet GUI should work with php4, but is has not been tested.

4.1 Remove PHP4

Php5 can also be installed with the –U parameter (upgrade), but this method was used in the tests.

- rpm -ev php-ldap
- rpm -ev --nodeps php-pear
- rpm -ev php

Note the “--nodeps” for php-pear. Php-pear is needed by php, so the parameter is needed to disregard dependencies.

4.2 Install PHP5

- cd php5
- rpm -ivh php-common-5.1.6-3.el4s1.10.i386.rpm
- rpm -ivh php-cli-5.1.6-3.el4s1.10.i386.rpm

- rpm -ivh php-5.1.6-3.el4s1.10.i386.rpm

Check installation:

- php –version

Should say “PHP 5.1.6 ...”

4.3 Install PHP ssh support

The ssh PHP extension is required by Napet GUI to control masterCLI. If there is an error saying package is already installed, you may proceed. The error can also be that there is a conflict with another package.

4.3.1 Dependencies

- cd php_ssh
- rpm –ivh autoconf-2.59-5.noarch.rpm
- rpm –ivh automake-1.9.2-3.noarch.rpm
- rpm –ivh openssl-0.9.7a-43.17.el4_8.6.i386.rpm
- rpm -ivh php-devel-5.1.6-3.el4s1.10.i386.rpm

4.3.2 libssh2

- gunzip libssh2-1.3.0.tar.gz
- tar -xvf libssh2-1.3.0.tar
- cd libssh2-1.3.0
- ./configure && make all install
- cd ..

4.3.3 libssh2 PHP extension

You might want to get the newest version from “http://pecl.php.net/package/ssh2”

- gunzip ssh2-0.11.3.tgz
- tar –xvf ssh2-0.11.3.tar
- cd ssh2-0.11.3
- phpize && ./configure -with-ssh2 && make
- cp ./modules/ssh2.so /usr/lib/php/modules/ssh2.so
- echo extension=ssh2.so >> /etc/php.ini
- service httpd restart

5 Configure connection settings

Napet Gui uses ssh to connect and for that it needs an ip address, a port number, a username and a password. These settings can be changes manually, but this can be done more easily with the NapetGUIConfig.exe. At this time, NapetGUIConfig is only for windows.

- Open the NapetGUIConfig -folder
- Start NapetGUIConfig.exe
- Press the Browse -button and open the functions.php -file from the Napet_GUI -folder
- Edit the ip, port, username and password values
- Press save
- Close NapetGUIConfig

6 Copy Napet GUI files

Copy all the files from “Napet_GUI” -folder to “/var/www/HTML/” on the server

7 Try it out

Navigate with your browser to the ip address of the server where Napet GUI was just installed. If you get a text saying “function ssh2_connect doesn't exist”, that means that the php and apache(httpd) are working correctly, but the ssh extension for PHP is not installed correctly.

NOTE: Do not use Internet Explorer

8 Problems? Optional way to install Napet GUI

When napet gui is installed on rhel 4, this causes sometimes an error message saying that masterCLI is slow, stuck or not running. Even though most of the times this can fixed by reloading the page, it is not something you want to see. Rhel 4 support was ended in 2008 and the ssh extension for php is not the best out there for rhel 4.
Originally napet gui was supposed to be run on a newer Linux version, where the ssh support is much better. E.g. on Fedora or Ubuntu the program would run much better. If you have a computer that has Fedora or Ubuntu (works also on virtual machines), then this could be the trick for you!

8.1 Installing packages

8.1.1 Ubuntu

- sudo apt-get install apache2 #installs the web server
- sudo apt-get install php5 #installs php5 support
- sudo apt-get install php5-cli
- sudo apt-get install libssh2-1-dev #installs ssh extensions development packages
- sudo apt-get install libssh2-php #installs ssh extension
- sudo service apache2 restart #restarts the web server

Configure the napet gui files, just like in chapter 5.
Copy files to your "/var/www/" on ubuntu.
Test the gui, open in ubuntu e.g. firefox and open the site "localhost". From other computers you can open this page by opening in the browser the ip-address of the Ubuntu machine.

11.1.1 8.1.2 Fedora

- sudo yum install httpd #installs the web server
- sudo yum install php #installs php5 support
- sudo yum install php-pecl-ssh2 #installs ssh extension
- sudo service httpd restart #restarts the web server

Configure the napet gui files, just like in chapter 5.
Copy files to your "/var/www/HTML/" on fedora.

Test the gui, open in fedora e.g. firefox and open the site “localhost”. From other computers you can open this page by opening in the browser the ip-address of the Fedora machine.

NOTE: Fedora might give an error, press “show” on the alert and follow instructions on how to allow this. (Press troubleshoot, follow instructions where to allow httpd connect through any port).

Or just type in the terminal “sudo setsebool -P httpd_can_network_connect 1”.

Now it should work.